COBB
INSIDE DOS™
Tips & techniques for MS-DOS & PC-DOS
November 1991
Vol. 2 No. 11

DOS 5's Shell offers an easier way to move files

If you've spent a lot of time working with DOS, you probably navigate from the command prompt instead of the Shell. However, the Shell brings more than just a pretty face to DOS because along with its graphical interface, the Shell provides some commands that aren't available at the prompt. One especially useful file-handling command is the File menu's Move... command. This command allows you to quickly accomplish a task that would be more cumbersome at the prompt. In this article, we'll take a look at moving files in the Shell.

Moving files in the Shell

As you probably know, moving files at the prompt takes two commands. First, you must copy the file to its new destination, and then you have to delete the old file. For example, to move the file DARRIN.DOC from the \YORK subdirectory to the \SARGENT subdirectory, you'd first type

```
C:\YORK>copy darrin.doc \sargent
```

and press [Enter] to copy the file. Then, you'd type

```
C:\YORK>del darrin.doc
```

to remove the file from the \YORK subdirectory.

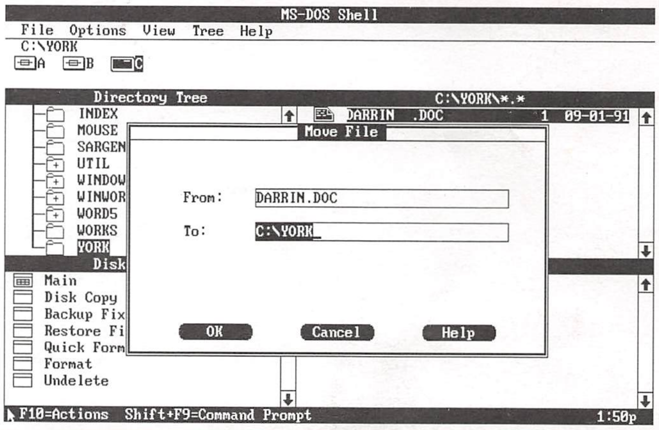
Using the Shell, however you can accomplish the task with only one command. First, we'll show you how to move the file using the Move... command. Then, we'll show you a shortcut you can use if you have a mouse.

The Move... command

The Shell's File menu offers a Move... command that is the equivalent of the COPY and DEL operation shown above. Using the Move... command, you can quickly move a file to another directory. To take advantage of this command, first you select the file or files you want to move in the file list window. (For the ins and outs of selecting files in the Shell, see the article "Managing Files with DOS 5's Shell" on page 9 of this issue.) Then, you pull down the File menu and choose the Move... command or press [F7]. When you do, DOS will present the Move File dialog box shown in Figure A. Simply type the name of the destination subdirectory and press [Enter] or click OK. Immediately, DOS will carry out the command.

Let's use this technique to move the DARRIN.DOC file from the \YORK subdirectory to the \SARGENT subdirectory. To begin, highlight the \YORK subdirectory in the Directory Tree window, then select the DARRIN.DOC filename in the file list window. Next, pull down the file menu and choose the Move... command (or press [F7]). When DOS presents the Move File dialog box, type *\SARGENT* in the dialog box's *To:* text box. (The *To:* text box will contain the source directory until you type in the name of the target directory.) Finally, press [Enter] or click OK. To confirm the move, select the \SARGENT subdirectory. DOS will display the file, as shown in Figure B on page 12.

Figure A



```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\YORK
 A  B  C

          Directory Tree                       C:\YORK\*.*
      INDEX                          DARRIN   .DOC          1  09-01-91
      MOUSE
      SARGEN            Move File
      UTIL
      WINDOW
      WINWOR       From:     DARRIN.DOC
      WORD5
      WORKS        To:       C:\YORK_
      YORK
            Disk
 Main
 Disk Copy
 Backup Fix
 Restore Fi
 Quick Form        OK          Cancel        Help
 Format
 Undelete
 F10=Actions  Shift+F9=Command Prompt                              1:50p
```

The DOS 5 Shell lets you move files with only one command.

Continued on back page

IN THIS ISSUE

INSIDE DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices	Domestic .... $59/yr. ($7.00 each) Outside the US .... $79/yr. ($8.50 each)
Address	The Cobb Group 9420 Bunsen Parkway, Suite 300 Louisville, KY 40220
Phone	Toll-free .... (800) 223-8720 Local .... (502) 491-1900 FAX .... (502) 491-4200
Staff	Editor-in-Chief .... Jim Welp Consulting Editors .... Mark W. Crane Tim Landgrave Contributing Editor .... Van Wolverton Editing .... Clyde Zellers Polly Blakemore Production .... Debbie Lane Design .... Karl Feige Publications Manager .... Elayne Noltemeyer Publisher .... Douglas Cobb

Address correspondence and special requests to The Editor, *Inside DOS*, at the address above. Address subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, at the address above.

Postmaster: Send address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

Conventions

To avoid confusion, we would like to explain a few of the conventions used in *Inside DOS*.

When we instruct you to type something, those characters usually appear on a separate line along with the DOS prompt. The characters you type will appear in red, while the characters DOS displays will appear in black.

Occasionally, we won't display the command you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "Issue the command *dir *.txt* at the DOS prompt." Although DOS is not case-sensitive, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is

copy *file1 file2*

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys, such as [Shift], [Ctrl], and [F1], appear in brackets. When two keys must be pressed simultaneously, those key names appear side by side, as in [Ctrl][Break] or [Ctrl]z.

LETTERS

Using the FOR command in Doskey macros

As you know, DOS' FOR command, which allows you to issue several commands at once, has slightly different syntax depending on whether you use it in a batch file or on the command line. When you use the FOR command in a batch file, it takes the form

```
for %%p in (set) do command
```

However, when you issue it from the prompt you drop one of the percent signs:

```
for %p in (set) do command
```

This difference in syntax can cause problems when you convert your batch files to Doskey macros. If you define your Doskey macros from the command line, you'll need to use only one percent sign in your FOR command. However, if you store your Doskey macro definitions in a batch file, you'll need to add the second percent sign. I learned this the hard way after I converted a batch file to a macro, then spent a long time trying to figure out what DOS meant by the arcane *Syntax error* message.

Luke Walter
Bridgeport, Connecticut

Mr. Walter's discovery can save other readers a lot of time—because DOS' error message isn't very clear, you might pull your hair out trying to debug your macros that use the FOR command.

The syntax problem comes into play because most users don't issue the FOR command at the command prompt. Instead, they typically use FOR in batch files and, for that reason, aren't aware of the difference in the command's syntax. However, with the advent of DOS 5 and Doskey macros, you need to become aware of the difference.

For example, to define a Doskey macro named TYPE that allows you to display the contents of several files with one command, you'd type

```
C:\>doskey type=for %i in ($*) do type %i
```

at the DOS prompt. However, if you store the macro's definition in a batch file, you'd use the line

```
doskey type=for %%i in ($*) do type %%i
```

For more information on how to create Doskey macros and store them in a batch file, see the articles "Doskey Macro's Put Complex Commands at Your Fingertips" and "Storing Your Macros and Automatically Loading them at Bootup" in the September 1991 issue of *Inside DOS*.

Shifting parameters in a batch file

In last month's issue of *Inside DOS*, we looked at DOS' GOTO and IF batch file commands in the article "Making Your Batch Files More Flexible With GOTO." As we demonstrated in that article, GOTO adds a lot of power to your batch files by enabling you to tell DOS to branch to another location in the batch file. This month, we'll first say a few words about replaceable parameters, and then we'll take a look at another batch file command, SHIFT. The SHIFT command allows you to shift parameters in a batch file. This technique provides some interesting possibilities by helping you overcome a few DOS command-line limitations.

Once you become comfortable with IF, GOTO, and SHIFT, you'll be able to write batch files that are more powerful and more versatile. Also, in future issues of *Inside DOS*, we'll present some batch files that take advantage of these commands.

Using replaceable parameters

As you probably know, most DOS commands require that you specify some data on which to operate. For example, the COPY command, which takes the form

```
copy source target
```

requires two pieces of data: *source* and *target*. Each time you issue the COPY command, you'll specify a different *source* and *target* on which it will operate.

Just as you need to supply some data for most of your DOS commands, you'll often want to supply data for your custom batch commands. Fortunately, DOS allows you to include a special symbol in your batch file called a *replaceable parameter*. When you execute the batch file, DOS will replace this symbol with the parameter you supply on the command line immediately following the batch command name.

The symbol you use as a replaceable parameter is a percent sign (%), followed by a number from one to nine. For example if you need to use only one replaceable parameter, use the symbol *%1*. If you need to use multiple replaceable parameters, use the symbol *%1* to represent the first parameter, *%2* to represent the second parameter, and so forth. For example, suppose you've created a batch file named ECHOALL.BAT that contains the following statements:

```
@echo off
echo %1
echo %2
echo %3
```

Now, if you issue the command *echoall*, followed by *party*, *on*, and *Wayne*, like this:

```
C:\>echoall party on Wayne
```

DOS will respond

```
party
on
Wayne
C:\>_
```

Now let's take a look at how the SHIFT command affects the way DOS handles replaceable parameters.

Moving parameters with SHIFT

As we mentioned, the SHIFT command shifts replaceable parameters in a batch file. When you include SHIFT in a batch file that calls for replaceable parameters, DOS shifts the parameters one position to the left. In other words, when DOS encounters the SHIFT command, it replaces the

An application for SHIFT: "complete delete"

In the article "Shifting Parameters in a Batch File," we showed you a batch file named ROTATE.BAT. Unfortunately, besides illustrating the SHIFT command, the ROTATE.BAT batch file doesn't really have a practical application. However, you can add usefulness to the batch file by changing only one word. Figure A shows a batch file, COMPDEL.BAT, that allows you to delete a group of files at the command line. As you can see, COMPDEL.BAT is identical to ROTATE.BAT, except for one word; instead of ECHO, we substituted the DEL command.

Figure A

```
@echo off

:START
if "%1"=="" goto :END
del %1
shift
goto :START

:END
```

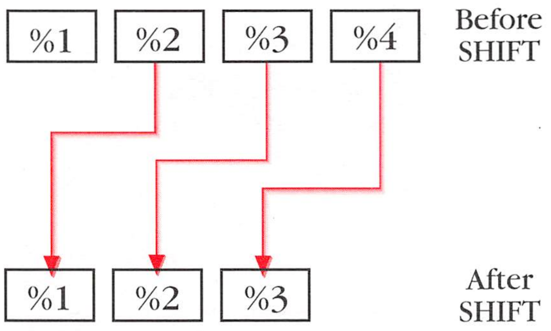
COMPDEL.BAT lets you delete a group of files with one command.

Of course, this batch file works the same as the original one, except instead of echoing your parameters to the screen, it deletes them. For example, instead of issuing three commands to delete the files CLYDE.TXT, POLLY.DOC, and DEBBIE.WK1, you could simply type

```
C:\DATA>compdel clyde.txt polly.doc debbie.wk1
```

parameter %1 with %2, %2 with %3, and so on. As DOS moves the parameters to the left, the parameter that was stored in %1 is deleted. Figure A illustrates how SHIFT moves the parameters. Using SHIFT, you can write batch files instructing DOS to carry out a command on all the parameters you specify until the list of parameters is exhausted.

Figure A



The SHIFT command shifts the parameters you type at the command line.

An example: ROTATE.BAT

For example, take a look at the batch file shown in Figure B. This batch file, called ROTATE.BAT, simply echoes to the screen the parameters you specify at the prompt.

Specifying more than nine parameters

In the article "Shifting Parameters in a Batch File," we mentioned that DOS allows you to designate the symbols %1 through %9 as replaceable parameters. Although nine replaceable parameters will handle most of your needs, you occasionally will want to specify more than nine parameters. As you may have already guessed, the SHIFT command allows you to get around this limitation. Because the ROTATE.BAT batch file we created in that article creates a loop that continues until all the parameters are exhausted (in effect, making each parameter, in turn, %1), you can specify as many parameters as you can squash into the 127 characters DOS allows on the command line. To test this, type

```
C:\BATCH>rotate a b c d e f g h i j k l m n o
p q r s t u v w x y z
```

DOS will echo each letter, one per line, before returning you to the prompt.

Figure B

```
@echo off

:START
if "%1"=="" goto :END
echo %1
shift
goto :START

:END
```

ROTATE.BAT uses the SHIFT command to echo to the screen the parameters you specify.

The meat of ROTATE.BAT is a section labelled :START. When DOS encounters this section of the batch file, it looks at the parameters the user specified at the command prompt and echoes them to the screen one at a time until it echoes the last parameter. In order to accomplish this feat, the batch file takes advantage of the IF and GOTO commands, which we discussed last month (in the article "Make Your Batch Files More Flexible with GOTO").

Here's how it works: The batch file's first command

```
@echo off
```

turns off echo, so that DOS doesn't display the commands onscreen as it carries them out. Next, the command

```
if "%1"=="" goto :END
```

tells DOS to end the batch file if there are no more replaceable parameters. In plain language, the IF statement says, "If there is no replaceable parameter (as specified by %1), skip over the ensuing commands until you reach the label :END." Since the label :END is the last line of the batch file, the batch file automatically stops executing there.

If there are replaceable parameters, DOS will proceed to the next command

```
echo %1
```

which echoes to the screen the text specified by the replaceable parameter %1. Next, the command

```
shift
```

tells DOS to move all the parameters specified on the command line one position to the left. Finally, the command

```
goto :START
```

tells DOS to branch to the label :START, which begins the process all over again. DOS will then continue in this loop until it has echoed all the parameters to the screen. Once it has exhausted all the parameters, the IF statement will cause DOS to exit the loop and end the batch file.

To run the batch file, type its name, *rotate*, followed by the parameters you want to echo to the screen. For example, if you specify the parameters *It's, party, time, it's*, and *excellent*, DOS will respond

```
C:\BATCH>rotate It's party time it's excellent
It's
party
time
it's
excellent
C:\BATCH>_
```

When DOS encountered the IF statement (which checks for command line parameters) it ignored the GOTO command because you specified parameters along with the command *rotate*. Next, it echoed the first parameter, *It's*. Then, it carried out the SHIFT command, which shifted all the parameters one position to the left, deleting the first parameter. Once DOS shifted the parameters, it carried out the GOTO command, returning to the beginning of the section labelled :START. The second time it encountered the ECHO command, it displayed the second parameter (now in the first position), *party*. It continued in this fashion until it echoed the last parameter, *excellent*.

Conclusion

DOS provides several commands that allow you to make your batch files more versatile. In this article, we looked at the SHIFT command, which lets you shift replaceable parameters. With SHIFT, you can write batch files that will accomplish what would otherwise require several commands.

VAN WOLVERTON

Take advantage of AUTOEXEC.BAT to customize your system

By Van Wolverton

Unless you've been using the same version of DOS and the same application programs for several years, you probably see periodic references to a file named AUTOEXEC.BAT. Application programs often offer to modify the file for you—although some ill-behaved programs change AUTOEXEC.BAT without giving you a choice—and starting with version 4, the DOS system disk itself included an AUTOEXEC.BAT file.

It's tempting to ignore AUTOEXEC.BAT if everything seems to be working alright, but it's worth a few minutes to examine the file and, if necessary, make a few changes. At a minimum, you might improve the performance of your system or save yourself a few minutes each day; in rare cases, you may *have* to change the file in order to use a program or new device.

And if you're using version 5 of DOS, you'll definitely want to to change AUTOEXEC.BAT to take advantage of DOS 5's memory-management capabilities.

A special batch file

You've most likely encountered batch files already, perhaps a simple one that starts an application program or displays directory entries in a particular form. In general, a batch file is nothing more than a standard DOS text file whose extension is BAT, and that contains DOS commands. When you type the name of the file as a command, DOS carries out the commands in the batch file as if you typed them.

You can create a batch file with most word processing or text editing programs, as long as the program doesn't put any special formatting characters in the file. You can use Edlin, the text editor that comes with DOS through version 4; the DOS Editor introduced in version 5 is an excellent tool for creating and revising batch files.

AUTOEXEC.BAT is like any other batch file, but with one exception: Each time you turn on your machine or restart DOS by pressing [Ctrl][Alt][Del], DOS looks in the root directory of the system disk for a file named AUTOEXEC.BAT; if it finds such a file, DOS automatically executes (hence *autoexec*) the commands in the file.

(Before looking for AUTOEXEC.BAT, DOS checks for another file in the root directory of the system disk, called CONFIG.SYS, which also contains commands to be carried out as part of the startup procedure. The commands in CONFIG.SYS, however, aren't standard DOS commands, like the ones in AUTOEXEC.BAT; they're special commands called *configuration commands* that tell DOS what programs are required to manage the devices attached to the system and how to manage the system memory.)

More important than ever

The importance of AUTOEXEC.BAT has increased significantly as DOS has matured. For the first few years—through version 2—it was used primarily for convenience, to automate a command or sequence of commands that were typed each time the system was started.

But as new devices became available and program developers began to take more advantage of DOS' capabilities, additional commands were required in AUTOEXEC.BAT. Most programs you install now make some sort of change to AUTOEXEC.BAT, if only to put the name of the program's directory in your command path; many devices you install also require changes to AUTOEXEC.BAT. Many programs and devices come with installation programs that try to be helpful by changing AUTOEXEC.BAT for you, but this can be a mixed blessing: Unless the installation program is intelligent as well as helpful, it can do more harm than good.

It's still possible, for example, to install a new program, then discover when you restart your system that the new program doesn't work or your system doesn't work the way it did before. One possibility: If the last command in your AUTOEXEC.BAT file starts a menu program, and an installation program adds a command at the end of AUTOEXEC.BAT, the new program or device probably won't work properly because DOS doesn't carry out the command it added until after you exit the menu program (which you may never do). If an installation program adds a PATH command at the end of AUTOEXEC.BAT that points to a new directory, this new command path replaces any earlier PATH command in AUTOEXEC.BAT and you won't be able to use your programs from all directories.

What's in AUTOEXEC.BAT?

You can put any DOS command in AUTOEXEC.BAT. In addition to commands required by programs or devices, you should include PATH and PROMPT commands to tailor your system to your needs; we'll talk more about PATH and PROMPT in a moment. Also consider including programs that:

- Send configuration commands to your printer.
- Copy files to a RAM disk, if you use one.
- Start a special program (called a *device driver*) for devices such as a mouse or CD-ROM drive.
- Start any terminate-and-stay-resident programs (TSRs) such as Doskey.
- Change the directory to the one in which you usually start.
- Start a program you always use first, such as Windows or a shell program.

To check what's in AUTOEXEC.BAT, change to the root directory by typing *cd * if necessary and then display AUTOEXEC.BAT with the TYPE command:

```
C:\>type autoexec.bat
```

DOS responds by displaying the contents of AUTOEXEC.BAT. You should see something like the following:

```
@echo off
path c:\dos;c:\util;c:\word;c:\win3;c:\excel
prompt $d  $t$_$p$g
\win3\mouse.com
SET TEMP=C:\WIN386\TEMP
cd \win3
win
```

The SET command creates an environment variable named TEMP that tells Windows where to store the temporary files it creates; it's in capital letters because it was added to AUTOEXEC.BAT by the Setup program that installs Windows.

Because your AUTOEXEC.BAT file should include both a PATH and PROMPT command, let's look at each of these commands a bit more closely. However, first take this precaution: Create a copy of your AUTOEXEC.BAT file with the extension BAK or SAV. That way you can easily restore your original AUTOEXEC.BAT file should the need arise.

Defining a command path

When you type something other than the name of a built-in DOS command (such as *dir* or *copy*), DOS looks in the current directory for the command file. If DOS can't find a file with the name you typed, it responds with the all-too-familiar

```
Bad command or file name.
```

The PATH command tells DOS where to find a program file if it isn't in the current directory. You can use the PATH command to define a *command path* that includes the directories containing the DOS files and your application programs; this lets you use any DOS command or any application program from any directory.

If you don't define a command path, you must change to the directory that contains the program file you want each time you use it. Obviously, it saves time to define a command path; you can save even more time by putting the PATH command in AUTOEXEC.BAT, because then you won't have to type the PATH command each time you start your system.

The path shown above in the sample AUTOEXEC.BAT file tells DOS where to find the program files for DOS itself, utility programs, Microsoft Word, Windows 3, and Excel. You can include as many directories as you like in the path, up to the DOS limit of 127 characters for each command. DOS searches for directories in the order in which they appear in the path, so put the directories that contain the commands you use most frequently at the beginning of the path.

When you install a new application program, add its directory to the PATH command in AUTOEXEC.BAT. If the application's installation program offers to do it for you (or simply does it and tells you about it), check AUTOEXEC.BAT to make sure it added the directory to your existing PATH command and didn't just add a PATH command at the end. If it did add a PATH command, revise your existing PATH command to include the new directory and delete the command added by the installation program.

Defining your own system prompt

The system prompt is what DOS displays when it's waiting for you to type a command. Unless you specify otherwise, the system prompt is simply the drive letter followed by a greater than symbol; if you have a fixed disk, that's C>. The PROMPT command lets you design your own system prompt to be more informative and personal.

Your prompt can consist of any letters or numbers you like, plus several items of information such as the time, date, current directory, a new line, or several special characters (such as > or $). You specify the information or special character you want by including a *$* followed by a single letter as a parameter to the PROMPT command. Table A lists the parameters the PROMPT command accepts and the information each provides.

Table A

This code...	inserts...
$t	the current time
$d	the current date
$p	the current directory
$_	a new line
$g	a greater than sign
$n	the current drive
$v	the DOS version number
$l	a less than sign
$b	a vertical bar
$$	a dollar sign
$q	an equal sign
$h	a backspace

You can use these special codes in the PROMPT command to customize your system prompt.

(The following examples assume that the current drive is C and the current directory is \WORD\LETTERS.)

Since it's always useful to know where you are, the following PROMPT command gives the most information for the least effort (press the [Spacebar] after the *p* before you press [Enter] to leave a space between the end of the system prompt and the beginning of what you type):

```
C>prompt $p
```

DOS will display

```
C:\WORD\LETTERS _
```

It doesn't take much more to create a two-line prompt that includes the time and the date on the first line, and the current drive and directory on the second (there are two spaces after *$d*; again, press the [Spacebar] before pressing [Enter]):

```
C:\WORD\LETTERS prompt $d  $t$_$p
```

DOS will display

```
Tue 10-16-91   14:40:23.15
C:\WORD\LETTERS _
```

Take the time to design a system prompt that includes what you want, then put the PROMPT command in AUTOEXEC.BAT.

If you're using version 5

Version 5 adds only one significant command to your AUTOEXEC.BAT file, and it's really important: LOADHIGH lets you run device drivers and other terminate-and-stay-resident programs (TSRs) in high memory, making more standard memory available for your application programs. Because much of DOS itself can also run in high memory (see "Modifying CONFIG.SYS to Take Advantage of DOS 5's Memory Enhancements" in the September 1991 issue of *Inside DOS*), when you upgrade from an earlier version of DOS you'll gain as much as 50K of standard memory; this can make your system run much faster, and even let you use programs that might not run with less memory.

Let's return to the AUTOEXEC.BAT file shown earlier; notice that it contains the following command to start a device driver program for a mouse:

```
\win3\mouse.com
```

If you're using version 5, change this command to read:

```
loadhigh \win3\mouse.com
```

or simply

```
lh \win3\mouse.com
```

Everything will work as before, but the program will now run in high memory, freeing almost 15K of standard memory. In order to maximize memory available to application programs, change any similar commands in your existing AUTOEXEC.BAT file that start TSRs or device drivers.

You should also include a LOADHIGH command that starts Doskey. Doskey is a TSR included with DOS 5 that lets you recall commands typed earlier and create macros. Doskey is a valuable tool you can use every day. To tell DOS to load Doskey each time you boot up, add the following command to AUTOEXEC.BAT:

```
loadhigh doskey
```

Don't panic

Of course, it's possible to make an error in your AUTOEXEC.BAT file that makes your system work incorrectly, or may even prevent DOS from starting at all. If this happens, you can't start the computer the normal way and correct the error, because DOS runs AUTOEXEC.BAT each time you start or restart your system. But don't panic—all isn't lost; just put the DOS system diskette in drive A and restart the system. Then change your AUTOEXEC.BAT file to eliminate the error, remove the DOS diskette from drive A, and restart your system as you normally do.

Contributing editor Van Wolverton is the author of the best-selling books Running MS-DOS 5 *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, currently lives in Alberton, Montana.*

DOS 5 TECHNIQUE

Copying every file *except* those you specify

In last month's issue of *Inside DOS*, we showed you how to hide files with DOS 5's ATTRIB command. As we demonstrated in that article, you can issue the command

```
C:\>attrib +h filename
```

to turn on the file's hidden attribute. Conversely, the command

```
C:\>attrib -h filename
```

unhides the file. When you hide a file using ATTRIB, DOS won't display the filename in a directory listing or allow you to manipulate the file using most DOS commands. In addition to providing security from intruders, the ATTRIB command's H switch allows you to change the way you normally handle files. In this article, we'll present a batch file that harnesses this feature of ATTRIB.

Hide then copy: NOCOPY.BAT

As we mentioned, when you hide files with DOS 5's ATTRIB command, other DOS commands won't recognize the filename. For example, suppose you issue the command

```
C:\DATA>attrib +h payroll.wk1
```

to hide the PAYROLL.WK1 file. Then, you try to copy the PAYROLL.WK1 file to the diskette in drive A with the command

```
C:\DATA>copy payroll.wk1 a:
```

DOS will respond

```
File not found - PAYROLL.WK1
                0 file(s) copied
```

Using wildcards, you can take advantage of this attribute to copy only certain files. The trick is to first hide all the files you don't want to copy. Then, you copy all the files in the directory, and finally unhide the hidden files. Figure A shows a batch file called NOCOPY.BAT that lets you accomplish just that. You can create NOCOPY.BAT with a text editor such as Edlin, Edit, or any ASCII compatible word processor.

Figure A

```
@echo off
attrib +h %1
copy *.* %2
attrib -h %1
```

NOCOPY.BAT lets you copy every file except those you specify at the prompt.

Here's how it works: The first command turns off echo, suppressing the display of the commands as DOS carries them out. Next, the command

```
attrib +h %1
```

hides the file you specify as the first parameter when you run NOCOPY.BAT from the command line. When you use a wildcard to specify a group of files, DOS will hide all the files that meet the wildcard specification.

Then, the command

```
copy *.* %2
```

copies all the files in the directory that aren't hidden or system files (DOS automatically hides all system files). The second replaceable parameter, *%2*, designates the drive or directory to which you want to copy the files. Finally, the last command

```
attrib -h %1
```

restores all the hidden files.

To run NOCOPY.BAT, just type its name, followed first by the wildcard specification that designates the files you *don't* want to copy, then by the destination drive or directory.

One useful application for this technique is to hide those files with a BAK extension. Many applications automatically create backups of your files as you save them. Typically, the application will assign the backup files the BAK extension. While this can be a handy feature, you might not want to clutter your diskette with duplicate files.

To copy all the files to drive A except those with the BAK extension, type

```
C:\DATA>nocopy *.bak a:
```

This command tells DOS to run the NOCOPY.BAT batch file, hiding all files with the BAK extension, copying all the remaining files to drive A, and then unhiding the BAK files.

Notes

Note that this batch file won't affect the settings of the read-only or archive attributes—an important consideration if your backup program uses the ATTRIB command's archive attribute. In addition, if you use the ATTRIB command to turn on the system attribute for files you want to remain invisible at all times, this command won't affect them either; DOS won't allow the file's hidden attribute to be turned on or off until its system attribute is turned off. Also, keep in mind that this technique isn't limited to the COPY command. Because ATTRIB's H switch hides files from all DOS commands (except ATTRIB, of course), you can tailor the batch file for other uses. To change the batch file to utilize another command, just substitute that command's name for the word *copy* on line 3.

Conclusion

Using the DOS 5 ATTRIB command's H switch, you can hide files not only from prying eyes, but also from DOS commands. In this article, we presented a batch file that exploits this ATTRIB feature. Using NOCOPY.BAT, you can copy all files except those you specify.

DOS 5 SHELL TIPS

Managing files with DOS 5's Shell

The DOS 5 Shell program offers a few commands that aren't available at the prompt. Moreover, the Shell often provides file-handling capabilities that are much quicker than those the prompt offers. However, before you can take advantage of these Shell features, you need to know how to tell the Shell which files you want to manipulate. In this article, we'll cover a few file-handling basics for those of you who are new to the Shell's graphical style of computing.

Selecting files in the Shell

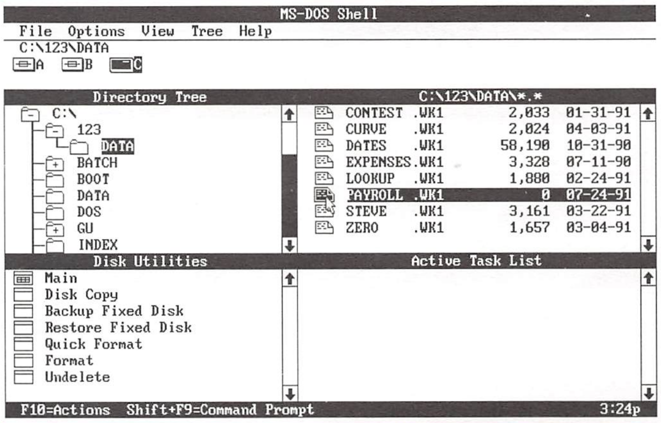
Once you've loaded the Shell, you can take advantage of the commands it offers. To load the Shell, type

```
C:\>dosshell
```

and press [Enter]. To instruct DOS to load the Shell at bootup, you can add the command *dosshell* to your AUTOEXEC.BAT file. (For more on AUTOEXEC.BAT, see Van Wolverton's column which begins on page 6.)

In order to take advantage of most Shell commands, you must first highlight or *select* a filename or group of filenames. Once you select a file, you can use the commands on the Shell's pull-down File menu to affect that file. To select a file using the mouse, first click on the name of the subdirectory that contains the file in the Directory Tree window. If the subdirectory you need isn't displayed in the Directory Tree window, click on the icon representing the directory that contains the subdirectory. Once you've selected the subdirectory, click on the filename in the file list window, as shown in Figure A. (By the way, if you don't see the directory or the filename in the window, you may need to scroll it into view. To do this, click on the bottom arrow of the scroll bar at the right side of either window.)

Figure A



```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\123\DATA
 A  B  C

          Directory Tree                          C:\123\DATA\*.*
 C:\                                  CONTEST  .WK1      2,033  01-31-91
   123                                CURVE    .WK1      2,024  04-03-91
     DATA                             DATES    .WK1     58,190  10-31-90
   BATCH                              EXPENSES.WK1       3,328  07-11-90
   BOOT                               LOOKUP   .WK1      1,880  02-24-91
   DATA                               PAYROLL  .WK1          0  07-24-91
   DOS                                STEVE    .WK1      3,161  03-22-91
   GU                                 ZERO     .WK1      1,657  03-04-91
   INDEX
          Disk Utilities                          Active Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
 F10=Actions  Shift+F9=Command Prompt                                3:24p
```

You can select a file by simply clicking on its filename in the file list window.

To select a file from the keyboard, first press [Tab] to activate the Directory Tree window. Next, press ↓ to select the directory containing the subdirectory that contains the file. Press the + ([Shift]=) key to open the subdirectory, then press the ↓ key to highlight the subdirectory itself. Notice that as you move through the list of subdirectories, DOS displays in the file list window the files they contain. Once you select the directory that contains the file you want to select, press [Tab] to activate the file list window. As soon as you do, DOS will highlight the first file in the list. Finally, press the ↓ and ↑ keys to select the files.

For example, let's use the keyboard to select the PAYROLL.WK1 file shown in Figure A. To begin, press the [Tab] key once to activate the Directory Tree window. Next, press ↓ once to highlight the \123 directory. Then, press + to display the \DATA subdirectory. Once you've displayed the \DATA subdirectory, press ↓ to highlight it. When you do, DOS will display in the file list window the contents of the \DATA subdirectory. To move to that window, press [Tab]. Finally, press ↓ five times to highlight the PAYROLL.WK1 file.

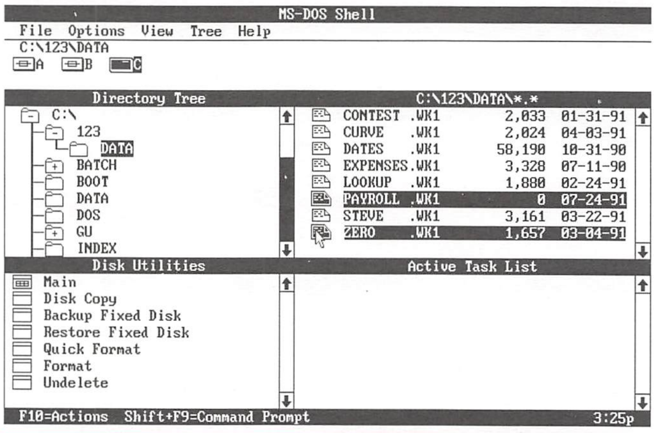
Selecting more than one file at once

Often, you'll want to select more than one file at a time. For instance, you might want to delete or move several files. To select more than one file with the mouse, just click once to select the first file, then hold down the [Ctrl] key and click on the second filename. DOS will highlight both filenames. Once you highlight the files and issue a command, DOS will carry out that command for both files. To select more files, continue with the [Ctrl]-click method. By the way, if you accidentally select a file, you can press [Ctrl] and click your mouse to deselect it without deselecting the rest of the files.

To select more than one file with the keyboard, begin by selecting the first file as outlined in the preceding section. Next, press [Shift][F8] to tell DOS you want to add another file to the selection. DOS will display *Add* on the status bar. Then, press the ↓ or ↑ key to move to the second filename, and press the [Spacebar] to select it. Continue pressing the arrow keys and the [Spacebar] until you've selected all the files you want to affect. Again, if you accidentally select a file you don't need to deselect the entire list—just move to the filename and press [Spacebar] while you're still in the Add mode. Finally, press [Shift][F8] to remove *Add* from the status bar.

Now, let's select a second file in the \DATA subdirectory using the mouse. With the PAYROLL.WK1 file still selected, hold down the [Ctrl] key and click on the ZERO.WK1 filename. DOS will add it to the selection, as shown in Figure B.

Figure B



```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\123\DATA
 A  B  C

        Directory Tree                         C:\123\DATA\*.*
 C:\                                 CONTEST .WK1      2,033  01-31-91
 ├─123                               CURVE   .WK1      2,024  04-03-91
 │ └─DATA                            DATES   .WK1     58,190  10-31-90
 ├─BATCH                             EXPENSES.WK1      3,328  07-11-90
 ├─BOOT                              LOOKUP  .WK1      1,880  02-24-91
 ├─DATA                              PAYROLL .WK1          0  07-24-91
 ├─DOS                               STEVE   .WK1      3,161  03-22-91
 ├─GU                                ZERO    .WK1      1,657  03-04-91
 ├─INDEX
        Disk Utilities                        Active Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
 F10=Actions  Shift+F9=Command Prompt                            3:25p
```

With the Shell, you can affect multiple files by selecting them before issuing commands.

Table A

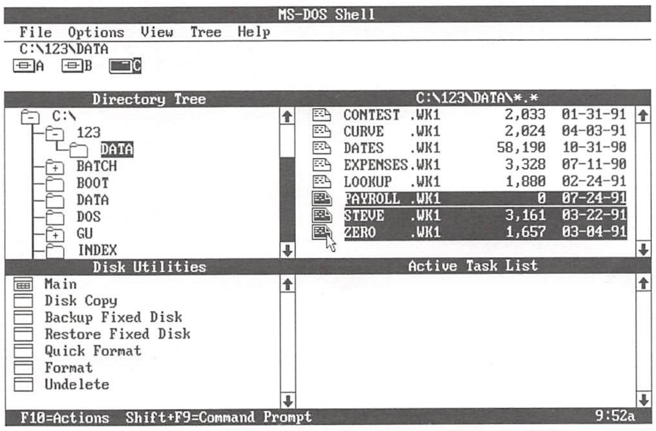
To. . .	use the mouse to. . .	or press. . .
Activate different windows in the Shell	click anywhere in the window	[Tab]
Select a file	click once on the filename	↓ or ↑
Select multiple files	click on the first file, then hold down [Ctrl] and click the other files	select first file, press [Shift][F8], then press ↓ or ↑ to move to other files followed by [Spacebar] to select them
Select files in sequence	click on the first file, then hold down [Shift] and click on the last file in the list	select first file, hold down [Shift], then press ↓ or ↑ to extend the selection

Here's how you select files in DOS 5's Shell.

Selecting files in sequence

If the files you need to select are sequential, you can save a couple of steps. To select a sequence of files using the mouse, click on the first filename in the sequence. Then, hold down the [Shift] key and click on the last filename in the sequence. DOS will highlight the two files and all the files between them, as shown in Figure C. To accomplish the same task using the keyboard, begin by selecting the first file in the sequence. Then, hold down the [Shift] key and press ↓ to extend the selection. Table A summarizes the steps you take to select files in the Shell.

Figure C



```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\123\DATA
 A  B  C

          Directory Tree                         C:\123\DATA\*.*
 C:\                                 CONTEST .WK1      2,033  01-31-91
   123                               CURVE   .WK1      2,024  04-03-91
     DATA                            DATES   .WK1     58,190  10-31-90
   BATCH                             EXPENSES.WK1      3,328  07-11-90
   BOOT                              LOOKUP  .WK1      1,880  02-24-91
   DATA                              PAYROLL .WK1          0  07-24-91
   DOS                               STEVE   .WK1      3,161  03-22-91
   GU                                ZERO    .WK1      1,657  03-04-91
   INDEX
          Disk Utilities                        Active Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
 F10=Actions  Shift+F9=Command Prompt                                 9:52a
```

You can use the [Shift] key in combingation with the ↑ and ↓ keys to quickly select a group of contiguous files.

Shortcuts for selecting files

Because selecting files is such a common task in the Shell, DOS includes a few shortcuts you can use. Table B lists the shortcuts available.

Table B

Key combination	Effect
Any letter	selects the first file that begins with that letter
[Home]	selects the first file in the directory
[End]	selects the last file in the directory
[Ctrl]/ (or the Select All command on the File menu)	selects all the files in a directory
[Ctrl]\ (or the Deselect All command on the File menu)	deselects all the files except the first selected file

Here are some shortcuts you can use to select files in DOS 5's Shell.

Conclusion

Using the Shell program, you can often manipulate files more easily than you can at the system prompt. However, before you can use the Shell's file handling commands, you must first *select* the file's name. In this article, we presented some tips for selecting files in the Shell.

SMARTDRV.SYS didn't make us look very smart

In the September 1991 issue of *Inside DOS*, we made a mistake in the article "Modifying CONFIG.SYS to Take Advantage of DOS 5's Memory Enhancements." In that article, we explained how you can set up a disk cache by placing the command

```
devicehigh=c:\dos\smartdrv.sys 2048 512
```

in your CONFIG.SYS file. We then pointed out that the command's first value (2048) specifies the size of the disk cache, and that the second value (512) specifies the sector size. Unfortunately, this information isn't correct.

The purpose of the first value in this command is exactly as we explained—it specifies the *initial* size of the disk cache (in kilobytes), and can range from 128 through 8192. In our case, then, the value 2048 sets up an initial cache size of 2 megabytes.

The second value, however, does not specify the sector size, as we explained in September. Instead, this value specifies the *minimum* size of the disk cache (also in kilobytes). The reason you'll want to specify a minimum cache size is that some programs (such as Microsoft Windows 3.0) can reduce the size of the cache and then use the memory formerly occupied by the cache for other purposes. In our case, then, specifying a minimum cache size of 512 kilobytes guarantees us that no programs can reduce the size of the cache to less than 512K.

We apologize for our oversight and for any confusion it may have caused you.

Microsoft Technical Support
(206) 454-2030

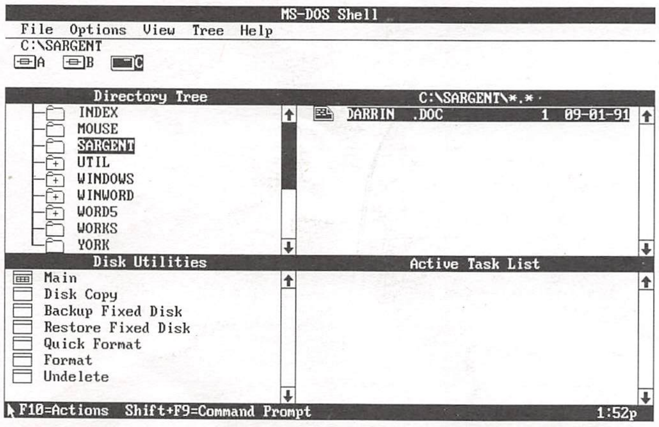
IMDC:1144914I:140692001 9211
FRANK RALLY
1796 GRACE AVE
SAN JOSE CA 95125-5620

Please include account number from label with any correspondence.

Continued from page 1

DOS 5's Shell offers an easier way to move files

Figure B



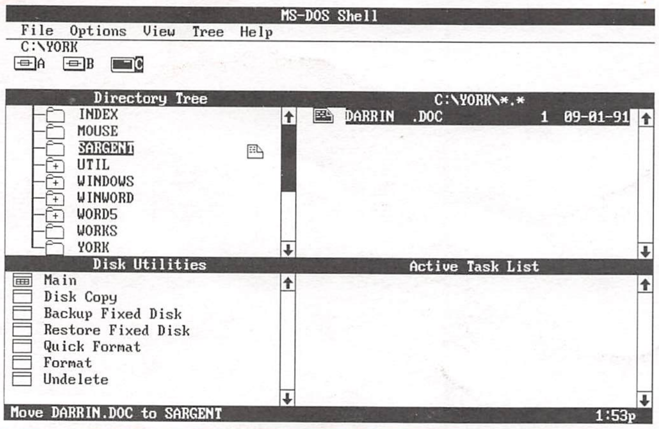
```
MS-DOS Shell
 File  Options  View  Tree  Help
  C:\SARGENT
 A  B  C
        Directory Tree                       C:\SARGENT\*.*
    INDEX                         DARRIN  .DOC        1  09-01-91
    MOUSE
    SARGENT
    UTIL
    WINDOWS
    WINWORD
    WORD5
    WORKS
    YORK
        Disk Utilities                        Active Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
F10=Actions  Shift+F9=Command Prompt                          1:52p
```

With one command, DOS copies files and deletes them from the source subdirectory.

Click and drag

If you use a mouse, moving files is even easier. You can move files among subdirectories by simply clicking on the filename (or names) and dragging it to the new location in the Directory Tree window. If the destination directory is out of view, you can display it by dragging the mouse pointer on top of the up or down scroll arrow until the directory scrolls into view. Let's again move the DARRIN.DOC file, except this time we'll use the mouse to drag the file to the new subdirectory.

Figure C

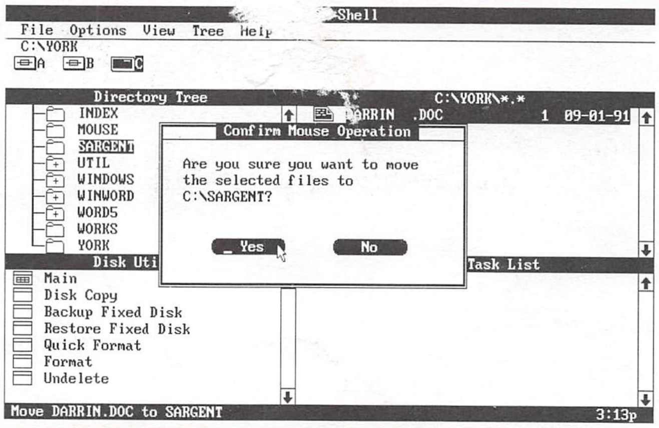


```
MS-DOS Shell
 File  Options  View  Tree  Help
  C:\YORK
 A  B  C
        Directory Tree                       C:\YORK\*.*
    INDEX                         DARRIN  .DOC        1  09-01-91
    MOUSE
    SARGENT
    UTIL
    WINDOWS
    WINWORD
    WORD5
    WORKS
    YORK
        Disk Utilities                        Active Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
Move DARRIN.DOC to SARGENT                                    1:53p
```

You can quickly move files in the Shell by clicking and dragging with the mouse.

To begin, move the mouse pointer to the DARRIN.DOC filename in the file list window and click and hold down the left mouse button. Then, drag the file to the \SARGENT subdirectory in the Directory Tree window, as shown in Figure C.

Figure D



```
Shell
 File  Options  View  Tree  Help
  C:\YORK
 A  B  C
        Directory Tree                       C:\YORK\*.*
    INDEX                         ARRIN  .DOC        1  09-01-91
    MOUSE           Confirm Mouse Operation
    SARGENT
    UTIL            Are you sure you want to move
    WINDOWS         the selected files to
    WINWORD         C:\SARGENT?
    WORD5
    WORKS
    YORK                 Yes           No
        Disk Uti                            Task List
 Main
 Disk Copy
 Backup Fixed Disk
 Restore Fixed Disk
 Quick Format
 Format
 Undelete
Move DARRIN.DOC to SARGENT                                    3:13p
```

DOS presents this dialog box to confirm the move operation.

As you drag the file into the Directory Tree window, DOS highlights the directories to help you determine your position onscreen. Just drag the file on top of the \SARGENT directory name until DOS highlights it, then release the mouse button. DOS will confirm the operation with the dialog box shown in Figure D. To confirm the operation, press [Enter] or click Yes.

[Ctrl]ing the Move... command

In the article "The Shell Offers an Easier Way to Move Files," we demonstrated how easy it is to move files using the mouse: You just click on the filename and drag it to the folder icon of the new directory. This move operation is the equivalent of the command prompt's COPY and DEL commands.

Of course, you won't always want to delete the file from the source directory. To that end, the Shell's File menu includes a Copy... command that works just like the prompt's COPY command. However, if you use a mouse, here's a quicker way: Hold down the [Ctrl] key and drag the file to the target directory's icon. When you hold down [Ctrl], DOS copies the file instead of moving it.